Tutorial 11:

Q1. Fill in the details to prove that $(AM)[k] \subseteq AM$.

Ans1: In $(AM)[k]$, boost probabilities so that probability of error is at most $2^{-|x|}$. Assume the 2nd last M sends $p(|x|)$ bits of information to A.

Basically, consider the interaction $(AM)[k-2]AMAM$, and status of the interaction (with respective probabilities of success/error in these paths) after $(AM)[k-2]A$.

(a) If the probability of error after the interaction is at least $2^{-|x|/2}$, we let it be considered to be 1 giving advantage to opponent. Note that there is at most $2^{-|x|/2}$ probability for such interaction paths, thus we have introduced error of at most $2^{-|x|/2}$ this way.

(b) For each of the other interaction paths, reduce probability of error to $2^{-p(|x|)-2}$ by usual boosting. Note that this boosting will not effect the interaction in the portion $(AM)[k-2]A$. Thus, the above boosting does not effect the communication by M after $(AM)[k-2]A$.

For paths in (b) now, use the $MAM \subseteq AM$ method to interchange M and A. Thus, the probability of error in each of these paths now is at most $1/4$.

Thus, total probability of error is at most $1/4 + 2^{-|x|/2} < 1/3$, and we have converted the interaction to $(AM)[k-2]AAMM = (AM)[k-1]$. Thus, we are done by induction.

Q2. Suppose SAT $\in PCP(r(n), 1)$ for some $r(n) = o(\log n)$. Then show that $P = NP$.

Ans2: $PCP(r(n))$ means we have $O(r(n))$ possible coin tosses, giving $2^{cr(n)}$ possible paths where at most some constant cq bits are queried, for some oonstant c. Now, $x \in L$, iff all the paths lead to acceptance (for some proof w given by the prover). This, can be written as a SAT formula of existance of a proof $p_1p_2p_3 \ldots p_m$, where $m \in O(q * 2^{c*r(n)})$ such that for each of the $2^{c*r(n)}$ possibilities of the coin tosses, a formula involving cq of the bits in $p_1p_2 \ldots p_m$ satisfy some condition. This is basically a SAT formula of length $O(q) * 2^{c*r(n)}$ using the bits $p_1, p_2 \ldots p_m$, and the question is whether there exist a proof $p_1p_2 \ldots p_m$ which satisfies the formula.

Thus, we have done a self-reduction of SAT, and using the previous tutorial we will get that SAT is in P.

Q3: Suppose L can be expressed as intersection of a language in NP, and a language in coNP. That is, there exists $L_1 \in NP$ and $L_2 \in coNP$, such that $L = L_1 \cap L_2$. Then show that $L \in PP$.

Ans1: Suppose $L = L_1 \cap \overline{L_2}$, where $L_1, L_2 \in NP$.

Then, $L = L_1 - (L_1 \cap L_2) = [L_1 - (L_1 \cap L_2)] \cup [(L_1 \cap L_2) - L_1]$, which is a symmetric difference of two NP languages. As $NP \subseteq PP$, L is a symmetric difference of two PP languages, and thus in PP based on result done in last tutorial.

For the following questions, let $A_M(x)$ and $R_M(x)$ denote the number of accepting paths and number of rejecting paths on input x by nondeterministic Turing Machine M. Let $D_M(x) = A_M(x) - R_M(x)$.

Q4. Suppose that M is a polynomial time bounded nondeterministic Turing machine and $[x \in L$ iff $D_M(x) > 0]$. Then show that $L \in PP$.

Ans 4: Extend the computation tree of M to be of the same depth (i.e., same number of coin tosses) on all paths, where each accepting path (rejecting path) of original M leads to 2 extra accepting paths (rejecting paths) compared to rejecting paths (accepting paths) of the new machine.

Above can be done as follows. Suppose the maximum depth (number of coin tosses) is q. Then, for each path of length $r \leq q$, extend it to a path of length $q+1$, where the machine repeats the accept/reject done after r tosses, if the next $q-r$ tosses are all heads; otherwise, it accepts/rejects iff the last toss is head/tail.

It is now easy to see that this winesses that $L \in PP$ (where probability of acceptance being 1/2 means rejection of input).

Q5. Suppose we are given two non-deterministic Turing machines M_1 and M_2.

(a) Show that one can construct a non-deterministic Turing machine M such that $D_M(x) = D_{M_1}(x) - D_{M_2}(x)$.

(b) Show that one can construct a non-deterministic Turing machine M such that $D_M(x) = D_{M_1}(x) * D_{M_2}(x)$.

Make sure that your M is polynomial time bounded, if M_1 and M_2 are polynomial time bounded.

Ans 5(a): M first tosses a coin. If it is heads, it follows M_1. If it is tails, it follows M_2 but switches answers of M_2 from accept to reject and reject to accept. Thus, the difference of accepting vs rejecting paths of M is the difference of accepting vs rejecting paths of M_1 minus the difference of accepting vs rejecting paths of M_2.

5(b): M runs M_1 and M_2 (with their respective tosses). M accepts if either both M_1 and M_2 accept or both reject. M rejects if one of M_1 and M_2 accepts and the other rejects. It is easy to verify that this works.

Q6. Suppose x and y are integers.

Let $P_n(x) = (x-1)\Pi_{i=1}^{n}(x-2^i)^2$.

Let $S_n(x) = \frac{P_n(-x)-P_n(x)}{P_n(-x)+P_n(x)}$.

Show that:

(a) If $1 \leq x \leq 2^n$, then $0 \leq 4P_n(x) < -P_n(-x)$.

(b) If $1 \leq x \leq 2^n$, then $1 \leq S_n(x) < 5/3$.

(c) If $-2^n \leq x \leq -1$, then $-5/3 < S_n(x) \leq -1$.

Below, $1 \leq |x|, |y| \leq 2^n$.

Let $A_n(x,y) = S_n(x) + S_n(y) - 1$. Show that

(d) if $1 \leq x \leq 2^n$ and $1 \leq y \leq 2^n$, then $A_n(x,y) > 0$.

(e) if $-2^n \leq x \leq -1$ or $-2^n \leq y \leq -1$, then $A_n(x,y) < 0$.

Use techniques of the above questions to show that PP is closed under intersection.

(a) Clearly, $P_n(x) \geq 0$ and $P_n(-x) < 0$. Note that $(x-2^i)^2 \leq (-x-2^i)^2$. Moreover, for $2^k \leq x \leq 2^{k+1}$, we have, $4(x-2^{k+1})^2 < (-x-2^{k+1})^2$. Part (a) follows.

(b) Case of $P_n(x)$ being zero is easy. For $P_n(x) > 0$, part (b) follows by simply putting $-P_n(-x) = (4+\delta)P_n(x)$, for $\delta > 0$, and simplifying, as numerator is $-(5+\delta)P_n(x)$ and denominator is $-(3+\delta)P_n(x)$ (when $P_n(x)$ is non zero).

(c) Similar to (b).

(d), (e): Follow easily from (b) and (c).

Now, suppose M_i is a PP machine which run in time p(length of input) and accept L_i, for $i = 1, 2$.

Now, $1 \leq D_{M_i}(z) \leq 2^{p(|z|)}$ if $z \in L$ and $-1 \geq D_{M_i}(z) \geq -2^{p(|z|)}$ if $z \notin L$ (using the appropriate version of PP definition).

Note that techniques of Q5, allow us to get any "polynomials" in $D_{M_i}(z)$, on input z, as a difference in accepting vs rejecting path by a polynomial time probabilistic Turing machine M.

Suppose, $A_n(x, y) = \frac{Q_1(x,y)}{Q_2(x,y)}$, for some polynomials Q_1 and Q_2 in x, y. Note that sign of $A_n(x, y)$ is the same as sign of $Q_1(x, y) * Q_2(x, y)$. So, we construct a PP machine M which has the difference of accepting vs rejecting paths as $Q_1(x, y) * Q_2(x, y)$, where $x = D_{M_1(z)}$ and $y = D_{M_2(z)}$ (note that M is polynomial time machine based on techniques of Q5). This would work as a PP machine for the intersection of languages accepted by M_1 and M_2 as it has more accepting paths compared to rejecting paths iff $A_n(x, y)$ is positive.

Q7. (a) SAT can be considered as a polynomial size circuit over the input variables using NOT gates and (2 input) AND/OR gates. Question is then whether some possible input leads to circuit giving 1 as answer.

Now, clearly QuadEQ is in NP as one can guess a value for the variables, and then check that all the quadratic equations are satisfied.

We reduce circuits (or SAT circuits if you wish) to QuadEQ by first labeling all the wires in the circuits as $u_1, u_2, \ldots, u_r$.

Then, output of AND gate with inputs u_i, u_j and output u_k can be written as: $u_i u_j - u_k = 0$.

Output of OR gate with inputs u_i, u_j and output u_k can be written as: $u_i + u_j - u_i u_j - u_k = 0$.

Output of NOT gate with input u_i and output u_k can be written as: $u_i + u_k = 1$.

Suppose the final output of the SAT circuit is u_k, then satisfiability of the formula can be written as: $u_k = 1$.

The input wires can be similarly written by equating the input x_i with the corresponding name of the wire.

It is easy to verify that the above quadratic equations are satisfied iff the SAT formula can be satisfied.